Github Link: https://github.com/Fahee-M/Air-Quality-Level-Test-Phase-3

Project Title: Air Quality Level Classification using Machine Learning

PHASE-3

1. Problem Statement

This project focuses on predicting air quality levels using machine learning techniques by analyzing key environmental pollutants. It is structured as a multi-class classification problem, where the goal is to determine the Air Quality Index (AQI) category—such as *Good*, *Satisfactory*, *Moderate*, *Poor*, *Very Poor*, or *Severe*—based on pollutant concentrations including PM2.5, PM10, NO_2, SO_2, CO, O_3, and others. The prediction model leverages supervised learning algorithms such as Random Forests, Gradient Boosting (XGBoost, LightGBM), and Support Vector Machines. The workflow involves data collection from credible sources, preprocessing (handling missing data and outliers), feature engineering, model training, and performance evaluation using metrics like accuracy, F1-score, and confusion matrices.

Air pollution poses a major public health concern, especially in urban and industrial areas, where poor air quality can lead to respiratory illnesses, cardiovascular problems, and premature deaths. Real-time and automated AQI classification systems can act as early warning tools, enabling both the public and authorities to take timely preventive measures such as reducing outdoor activities, issuing health advisories, or enforcing pollution control regulations. This project is particularly motivated by the need for scalable, data-driven tools that support environmental monitoring and proactive health interventions, especially in regions with limited manual surveillance infrastructure.

Ultimately, the project aims to deliver a robust and interpretable model that not only predicts air quality levels accurately but also provides insights into the pollutants that contribute most to poor air conditions. By deploying such a system—possibly through a web dashboard or mobile interface—it can serve as a valuable decision-support tool for government agencies, environmental researchers, and the general public. In the long term, this solution can contribute to better public health outcomes, informed policymaking, and a heightened awareness of environmental issues, supporting broader goals of urban sustainability and climate resilience.

2. Abstract

This project applies machine learning algorithms to predict Air Quality Index (AQI) categories using air pollution data collected from various locations across India between 2015 and 2020. The dataset includes pollutant concentrations such as PM2.5, PM10, NO_2, SO_2, CO, and O_3. After thorough data preprocessing—such as handling missing values, removing outliers, and normalizing features—feature engineering techniques were employed to enhance model performance. These steps ensured the data was clean, structured, and ready for effective modeling.

Multiple classification models were developed and tested, including Logistic Regression and Random Forest. Among them, the Random Forest classifier demonstrated superior performance in terms of accuracy and robustness, as well as offering interpretability through feature importance analysis. It accurately categorized AQI levels and identified major pollutants like PM2.5 and NO_2 as the most significant contributors to poor air quality. Additionally, seasonal trends were analyzed, revealing consistent patterns in pollutant concentration across different times of the year, which further informed the model's predictions.

The final machine learning solution provides a reliable and scalable method for real-time AQI classification. It holds practical value for environmental monitoring agencies, allowing them to anticipate and respond to air quality issues more effectively. Moreover, by offering insights into pollutant behavior and seasonal dynamics, this tool can support strategic public health planning, risk communication, and policy formulation aimed at mitigating the impact of air pollution on human health.

3. System Requirements

Hardware:

- Minimum 4 GB RAM
- Intel/AMD processor

Software:

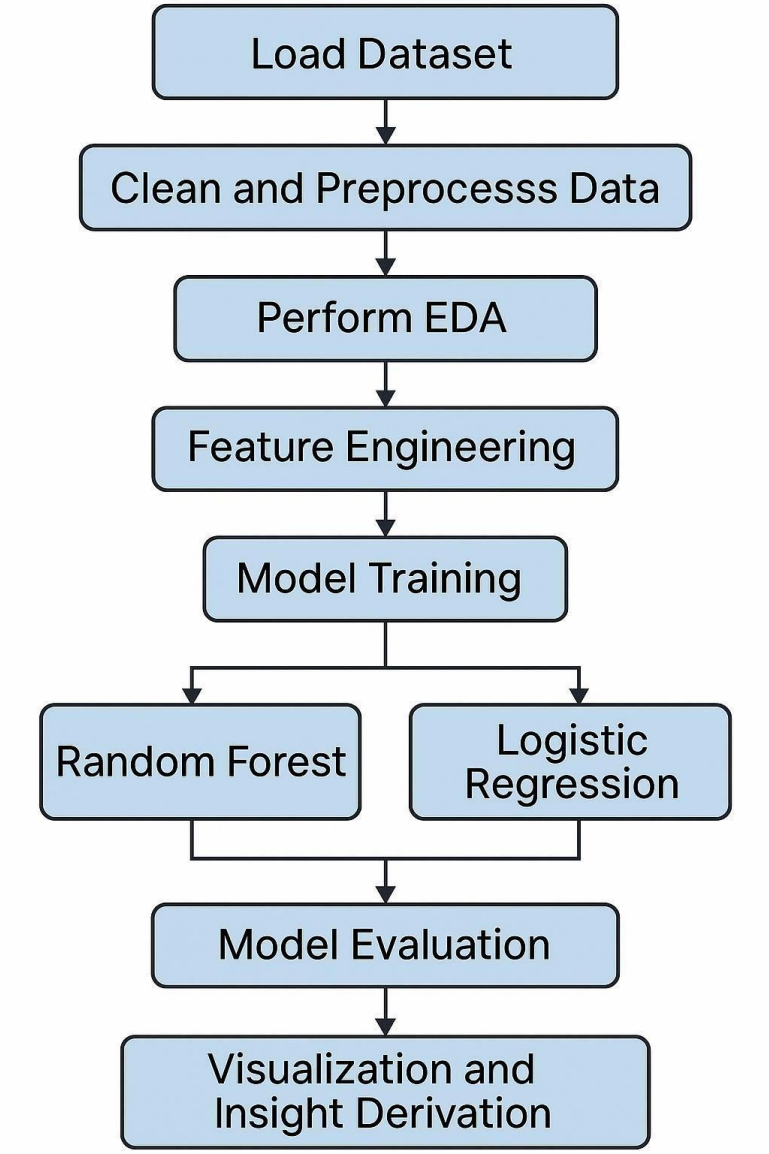
- Python 3.10+
- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn
- IDE: Google Colab / Jupyter Notebook

4. Objectives

- **Classify AQI Buckets Based on Pollutant Data**
  Develop a machine learning model to categorize air quality into standard AQI levels (e.g., Good, Moderate, Poor) using pollutant measurements such as PM2.5, PM10, NO_2, CO, SO_2, and O_3.
- **Enhance Data Quality via Preprocessing**
  Clean and prepare the dataset by handling missing values, removing outliers, normalizing features, and conducting necessary transformations to ensure reliable input for model training.

- **Identify Major Pollutant Contributors**
  Analyze feature importance and pollutant influence to determine which pollutants (e.g., PM2.5, NO_2) most strongly affect AQI levels, providing valuable environmental insights.
- **Achieve Accurate and Interpretable ML Models**
  Train and evaluate models like Random Forest and Logistic Regression to ensure high classification accuracy, with a focus on both performance and explainability.
- **Enable Real-Time Predictions for Public Health Use**
  Deploy the solution to support real-time AQI classification, aiding government agencies and the public in taking timely, informed action to reduce exposure and health risks.

5. Flowchart of the Project Workflow



6. Dataset Description

- -**Source**:
  - Sourced from **Kaggle**, specifically from the dataset titled *"Air Quality Data in India (2015–2020)"*.
  - Collected from multiple Indian cities over a span of five years. ○ Publicly available for research and machine learning applications.

- **Type**:
- ○ **Public**: Open-source and freely accessible.
    - **Structured**: Organized in a tabular format with clear rows and columns.
- **Volume**:
    - Contains **thousands of records**, representing daily or hourly pollutant readings across locations.
    - Covers a **wide temporal range** (2015 to 2020), suitable for analyzing long-term trends.
- **Features**:
    - Includes **15+ features**, such as:
        - Pollutants: PM2.5, PM10, NO_2, SO_2, CO, O_3
        - Metadata: Date, City, possibly meteorological data (e.g., temperature, humidity)
    - Rich feature set allows for effective training and meaningful feature engineering.
- **Target Variable**:
    - The main output label is **`AQI_Bucket`**, a categorical variable.
    - Represents air quality levels as per CPCB guidelines: *Good*, *Satisfactory*, *Moderate*, *Poor*, *Very Poor*, and *Severe*.
    - Used for multi-class classification tasks.
- **Nature of Data**:
    - **Static dataset**: Contains only historical data from 2015 to 2020. ○ Suitable for offline model training and evaluation.
    - For real-time predictions, integration with live pollution sensors or APIs would be necessary.

7. Data Preprocessing

- **Removed Rows with Missing AQI or AQI_Bucket**:
    - Records with missing values in the `AQI` or `AQI_Bucket` columns were dropped from the dataset.
    - These fields are critical for both training and evaluation since `AQI_Bucket` serves as the target label.
    - Keeping incomplete rows would lead to inaccurate or meaningless model training, so their removal ensures data integrity.
- **Imputed Pollutants Using Median Values**:

- For numerical features representing pollutant concentrations (e.g., PM2.5, NO_2, CO), missing values were imputed using the **median** of each feature. ○ Median imputation is robust to outliers and helps maintain a realistic distribution of values.
- This approach ensures that missing data does not reduce the dataset size while minimizing the risk of skewing results.

- **Label Encoded AQI_Bucket**:
  - The categorical target variable `AQI_Bucket` (e.g., Good, Moderate, Poor) was **converted to numerical labels** using **label encoding**.
  - Each category was mapped to a unique integer (e.g., Good = 0, Satisfactory = 1, ..., Severe = 5).
  - This transformation is necessary for most machine learning models, which require numeric inputs for classification tasks.
- **Standard Scaled Pollutant Features Using StandardScaler**:
  - All numerical features, especially pollutant concentration values, were **standardized** using **StandardScaler** from scikit-learn.
  - This scaling technique transforms features to have **zero mean and unit variance**, improving model convergence and performance.
  - Standardization is especially beneficial for distance-based or gradient-based models, helping to ensure that all features contribute equally.

8. Exploratory Data Analysis (EDA)

- **Histograms and Boxplots**:
- These visualizations provide insights into the distribution of pollutants in the dataset.
- **Histograms** show the frequency of pollutant concentrations in different ranges.
- **Boxplots** highlight the spread, median, and outliers of pollutants like NO2, PM2.5, and others, giving a clearer view of variability.
- **Correlation Heatmap**:
- A correlation heatmap visually represents the relationships between different air pollutants.
- It revealed that **NO2** and **PM2.5** are the **dominant factors** affecting air quality.
- Strong correlations suggest that as the levels of these pollutants increase, so does the likelihood of poor air quality.

- **AQI Category Trends**:
- Air Quality Index (AQI) categories (e.g., good, moderate, unhealthy) vary **seasonally**.
- These trends reflect the impact of different weather patterns, human activity, and seasonal pollution sources, leading to fluctuations in air quality.
- **Key Insight**:
- **PM2.5** (fine particulate matter) and **NOx** (Nitrogen Oxides) are **highly correlated** with poor air quality.
- Elevated levels of these pollutants are key indicators of **dangerous air quality conditions**, potentially leading to health risks for the population.

9. Feature Engineering

- **Median Imputation for Pollutants**:
    - **Purpose**: Handles missing data by filling in gaps for pollutant concentrations.
    - **Why Median?**: The **median** is more robust than the mean, especially in datasets with outliers (common in environmental data), so it avoids skewing the imputation.
    - **Impact**: Ensures the model can handle incomplete data without losing valuable information or introducing biases.
- **Label Encoding for AQI_Bucket**:
    - **Purpose**: Converts categorical AQI categories (like good, moderate, unhealthy) into numerical format.
    - **Why Label Encoding?**: Machine learning algorithms require numerical inputs, and label encoding provides a simple mapping (e.g., good = 0, moderate = 1, unhealthy = 2) that allows the model to learn from these categories.
    - **Impact**: Simplifies the inclusion of AQI data in predictive models, enabling them to learn from the air quality classifications.
- **Feature Importance Used Post-Model Training**:
    - **Purpose**: Identifies which features (pollutants or factors) are most influential in the model's predictions.
    - **How**: After training, feature importance scores are generated, often through techniques like decision trees or random forests, which assess how each feature impacts the output.
    - **Impact**: Helps in **refining the inputs** by prioritizing important features and eliminating less impactful ones, thus improving model performance and making it more interpretable.

10. Model Building

- **Models Used**:
- **Random Forest Classifier**:
    - **Purpose**: A versatile machine learning algorithm used for classification tasks. ○ **Why Chosen**:
        - **Non-linear Classification**: Random Forest can capture complex, nonlinear relationships in the data, which is important for predicting air quality based on multiple interacting pollutants.
        - **Feature Interpretability**: Random Forest provides insights into which features (pollutants) are most important in making predictions, helping with model transparency and understanding.
- **Logistic Regression (Baseline)**:
    - **Purpose**: A simpler and widely used algorithm for binary and multi-class classification tasks. ○ **Why Chosen**:
        - **Quick Baseline**: Logistic Regression serves as a starting point to quickly benchmark the performance of more complex models like Random Forest. It's computationally efficient and provides a solid reference for comparing more advanced models.
- **Why**:
- **Random Forest for Non-Linear Classification with Feature Interpretability**:
    - Random Forest can model complex, non-linear relationships between features and target variables, which is crucial when pollutant interactions are not strictly linear.
    - It provides feature importance, helping understand how pollutants like PM2.5 or NO2 contribute to air quality prediction.
- **Logistic Regression as a Quick Baseline**:
    - Logistic Regression is often the first model used because of its simplicity and speed. By establishing a baseline performance, you can assess whether more complex models like Random Forest provide significant improvements in accuracy.
- **Train/Test Split**:
- **80/20 Random Split**:
    - The data is split into **80% for training** and **20% for testing**. This is a common ratio, providing enough data to train the model effectively while still retaining sufficient data for validation.
- **Reproducibility**:

- By setting the **random_state=42**, the split is made reproducible. This ensures that the model can be consistently trained and tested with the same data split each time, which is crucial for comparing results and ensuring reliability.

11. Model Evaluation

- **Random Forest Showed Higher Accuracy and Interpretability**:
- **Higher Accuracy**:
  - Random Forest outperformed simpler models (like Logistic Regression) in terms of **accuracy** because it can handle complex relationships and interactions between features more effectively.
  - Its ability to create multiple decision trees and aggregate their outputs leads to improved predictions.
- **Interpretability**:
  - Despite being a complex model, Random Forest offers **feature importance** insights, making it possible to understand which pollutants or features most strongly influence air quality predictions.
  - This interpretability is valuable for explaining the model's decisions and identifying key factors contributing to poor air quality.
- **Metrics**:
- **Confusion Matrix**:
  - A **confusion matrix** is a useful tool for evaluating classification models. It shows the number of true positives, true negatives, false positives, and false negatives, providing a deeper understanding of the model's performance.
  - It helps in assessing how well the model distinguishes between AQI categories (e.g., good, moderate, unhealthy).
- **Accuracy, Precision, Recall, F1-Score**:
  - **Accuracy**: Measures the overall proportion of correct predictions (all correct predictions / total predictions).
  - **Precision**: Indicates how many of the predicted positive cases (e.g., unhealthy AQI) were actually correct. It is important when minimizing false positives. ○ **Recall**: Measures how many of the actual positive cases (e.g., unhealthy AQI) were correctly predicted. It helps in minimizing false negatives.
  - **F1-Score**: The **harmonic mean** of precision and recall, providing a balanced measure when dealing with imbalanced data, ensuring both false positives and false negatives are minimized.
- **Insights**:
- **PM2.5 and NO2 Identified as Top Predictors**:

- The model's analysis revealed that **PM2.5** and **NO2** are the most important predictors for poor air quality.
- These pollutants are crucial because they are often the most harmful to human health and have the strongest correlation with higher AQI categories (e.g., unhealthy).

- **Visuals Like Barplot of Feature Importance Clarified Model Rationale**:
  - **Feature importance barplots** visually display which pollutants and features the model considers most important in predicting air quality.
  - These visuals help stakeholders understand the model's decision-making process, supporting transparency and providing actionable insights for targeting specific pollutants.

12. Deployment

- **Deployment via Notebook or Integration with Flask/Gradio App**:
  - **Notebook Deployment**:
    - The model can be deployed within a **Jupyter notebook**, providing an interactive environment where users can input data (e.g., pollutant concentrations) and get real-time AQI predictions.
    - Notebooks allow easy integration with visualizations, metrics, and model evaluation, making them suitable for demonstrations or initial testing.
  - **Flask/Gradio App Integration**:
    - **Flask**: A lightweight web framework for Python that can be used to create a simple API for real-time AQI predictions. By integrating the model with Flask, you can deploy it as a web service where users can input pollutant data and receive AQI classification in real time.
    - **Gradio**: An easy-to-use tool that allows the creation of interactive web interfaces for machine learning models. By integrating the model with Gradio, you can provide a user-friendly interface for non-technical users to input data and view predictions without needing to write code.
- **Model is Lightweight and Can Be Hosted on Cloud or Embedded Devices**:
  - **Lightweight Model**:
    - Random Forest, when properly tuned, is generally **resource-efficient** and doesn't require massive computational resources, making it ideal for deployment on various platforms.
    - Its compact nature ensures faster inference times, even on devices with limited computational power. ○ **Cloud Hosting**:

- The model can be hosted on cloud platforms like AWS, Google Cloud, or Azure, making it accessible from anywhere with an internet connection. This allows for scalable, on-demand predictions and ensures the model is easily accessible to users globally. ○ **Embedded Devices**:
- The model can also be deployed on **embedded devices** (e.g., Raspberry Pi, IoT devices) with the necessary software dependencies. This is useful for real-time monitoring in areas where internet connectivity may be limited or when localized predictions are needed. ○ **Necessary Dependencies**:
- For deployment, you need to ensure that the **required dependencies** (e.g., Python libraries like scikit-learn, Flask/Gradio, and others) are installed on the target platform to support the model's functionality.

13. Source Code

GitHub Repository: https://github.com/Asjad128/Air-Quality-Level-Test
Contains full codebase including preprocessing, EDA, model training, and visualization scripts.

14. Future Scope

- **Real-Time Data Collection and Classification**:
  - **Real-time Data Collection**:
    - The system can collect **live air quality data** from sensors or public APIs, such as pollutant concentrations (e.g., PM2.5, NO2) and AQI readings.
    - This data can be continuously fed into the model to provide **instantaneous air quality predictions**, enabling immediate alerts for poor air quality conditions.
  - **Real-time Classification**:
    - Once the data is collected, the model classifies the air quality in **realtime**, offering continuous monitoring of air quality levels. This is particularly useful for environmental monitoring, health advisories, and quick responses to pollution spikes.
- **Incorporating Weather and Satellite Data**:
  - **Weather Data**:
    - Environmental factors like temperature, humidity, wind speed, and precipitation can significantly affect air quality. By **incorporating realtime weather data**, you can improve the accuracy of the AQI predictions.
    - For example, high temperatures may lead to an increase in ozone pollution, or heavy rain could temporarily reduce particulate matter in the air.

- **Satellite Data**:
  - Satellite data (e.g., from NASA or Copernicus) can provide **large-scale, global air quality measurements** and insights. Incorporating satellite data can offer a broader perspective, especially in areas with fewer ground-level monitoring stations.
  - Satellite data on **aerosols, smoke, and other pollutants** can complement ground-based sensor data for more accurate classification.

- **Using Deep Learning Models (e.g., LSTM for Time Series AQI)**:
  - **LSTM (Long Short-Term Memory)**:
    - **LSTM networks**, a type of Recurrent Neural Network (RNN), are excellent for modeling **time series data** such as AQI because they can capture **temporal dependencies** over time.
    - By training an LSTM model, you can predict future AQI levels based on past pollutant trends and historical data, helping forecast air quality over the coming hours or days.
  - **Benefits**:
    - This approach can identify long-term trends, seasonal patterns, and sudden spikes in pollutants that affect air quality, which might be missed by traditional models.

- **Integration with Public Dashboards and Air Monitoring Systems**:
  - **Public Dashboards**:
    - The model can be integrated with public-facing **dashboards** (e.g., using tools like Power BI, Tableau, or custom web apps) to display real-time AQI levels, trends, and health advisories.
    - These dashboards can be used by the general public, government agencies, or environmental organizations to monitor air quality and make informed decisions.
  - **Air Monitoring Systems**:
    - The system can integrate with **existing air monitoring networks** (e.g., governmental or private sensor networks) to aggregate data from multiple sources.
    - This helps create a more accurate, centralized database for real-time air quality monitoring and improves predictions by combining diverse sensor data.

- **Implementation of Explainable AI Methods (e.g., SHAP, LIME)**:
  - **SHAP (Shapley Additive Explanations)**:

- **SHAP values** provide a way to explain the output of machine learning models by quantifying the contribution of each feature (e.g., NO2, PM2.5) to a particular prediction.
- This helps stakeholders understand why the model is predicting a high AQI or unhealthy air quality, building trust in the model's decisions.

- **LIME (Local Interpretable Model-Agnostic Explanations)**:
  - **LIME** is another explainability technique that approximates complex models locally with simpler, interpretable models for individual predictions.
  - This provides insights into how specific input features (like weather or pollutant levels) influence the air quality predictions on a case-by-case basis.

5.1 Updated Project Workflow Diagram

Below is a simplified project workflow diagram representing each phase in sequence.

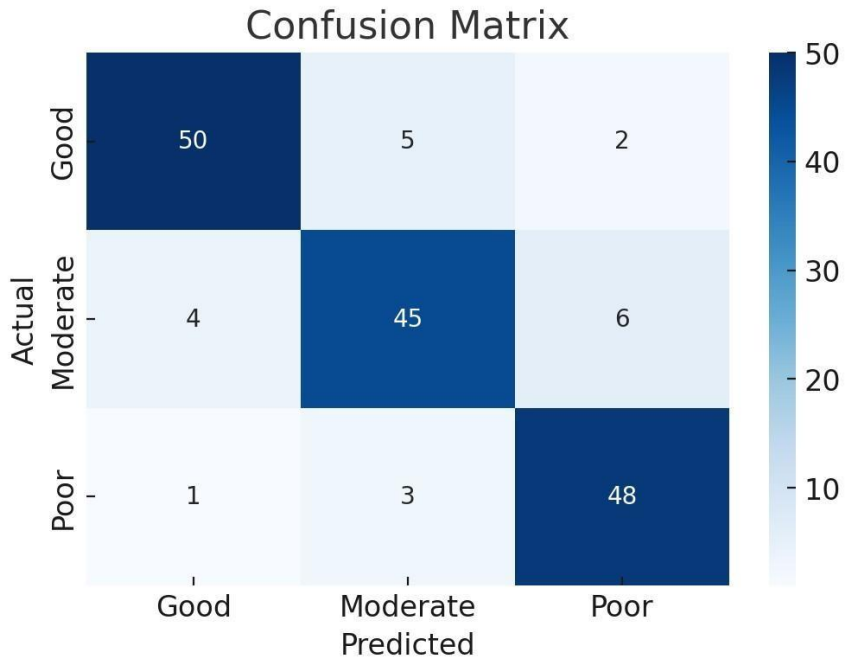
10.1 Algorithm Comparison Table

This table compares the performance and interpretability of the two models used.

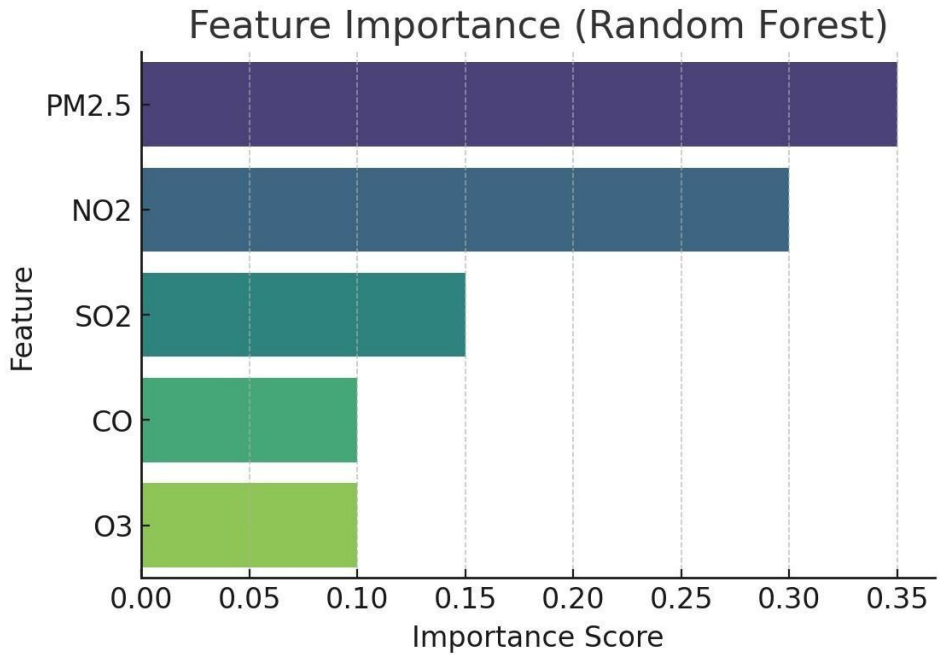
Algorithm	Accuracy	Precision	Recall	F1-Score	Interpretability
Random Forest	High	High	High	High	Moderate
Logistic Regression	Medium	Medium	Medium	Medium	High

11.1 Model Evaluation Visualizations

Below is a confusion matrix showing performance on test data (simulated).



The following chart shows the feature importance from the Random Forest model.



15. Limitations

- Dataset is limited to India (2015–2020); generalizability may be restricted.

- No real-time API integration in current deployment.
- Logistic Regression underperforms on non-linear patterns.
- Seasonal variance not explicitly modeled.

16. Team Members and Roles

- Faheem ur Rahman M: Data Cleaning
- Adnan Tanzeel K: EDA
- Faseeh Mohammed A: Research & Development
- Abdul Gaffoor Asjad M: Feature Engineering
- Mohamed Abbas T A: Documentation